

A Golay-Viterbi Concatenated Coding Scheme for MJS'77

L. D. Baumert and R. J. McEliece
Communications Systems Research Section

This article describes in some detail a proposed method of delivering the non-imaging science data on the Mariner Jupiter/Saturn (MJS) mission at a bit error probability which is substantially lower than the bit error probability required for the imaging data. The method is a "pre-coding" of the nonimaging data with an interleaved (24, 12) Golay code.

I. Introduction

It is well known that deep-space telemetry encoded with a (32, 6) biorthogonal block code and decoded with a maximum-likelihood decoder requires considerably less energy to achieve a given bit error probability than telemetry that has not been encoded. Since 1969, all of JPL's Mariner-class spacecraft have been equipped with such a coding system.

It is, however, also well known that short-constraint-length *convolutional* encoding, coupled with Viterbi decoding, is superior to biorthogonal coding. Thus, current plans call for the replacement of the biorthogonal code with a convolutional code on the Mariner Jupiter/Saturn 1977 and later JPL deep-space missions. This convolutional code will probably be either constraint

length 7, rate 1/2, or constraint length 6, rate 1/3. Table 1 contains the bit signal-to-noise ratio E_b/N_0 , required to obtain the bit error probabilities 0.01, 0.005, and 0.001 for uncoded, biorthogonally encoded, and convolutionally encoded telemetry.

(It is seen from Table 1 that the (6, 1/3) convolutional code is about 0.5 dB better than the (7, 1/2) code. However, future NASA deep-space telemetry will be transmitted at rates on the order of 10^5 bits/s, and at such rates the (6, 1/3) code will generate 3×10^5 symbols/s as compared to only 2×10^5 for the (7, 1/2) code. The extra bandwidth requirements and implementation difficulties of handling such high symbol rates may force the adoption of the (7, 1/2) code in spite of its theoretically inferior performance.)

As has been the case on previous Mariner-class missions, the minimum bit error probability required for the imaging data (i.e., the television coverage of the planets) on MJS is in the vicinity of 0.01–0.005. However, the nonimaging data turn out to be much more susceptible to noise than the TV. Final minimum bit error probabilities required by the nonimaging scientific experiments have not yet been set, but a commonly quoted upper bound is 5×10^{-5} .

Since the TV data will comprise over 95% of the total MJS telemetry, it would obviously be wasteful to deliver the entire telemetry stream at a bit error probability of 5×10^{-5} ; in fact, it would require about 1.6 dB more energy per transmitted bit to accomplish this. In this article, we will document one feasible method of delivering the nonimaging MJS telemetry at the lower bit error probability at a cost of only about 0.2 dB in signal power. This is the method of *concatenated*, or *nested*, coding. Specifically, the scheme we propose is the concatenation of the short-constraint-length convolutional code with an interleaved (24, 12) Golay code.

It should be mentioned that there are at least two other possible ways of efficiently delivering the nonimaging data more reliably than the TV. The first is the use of dual-channel telemetry, wherein the imaging and nonimaging data streams are encoded independently and frequency-multiplexed together. The second is the use of *lengthened-symbol* encoding, in which the nonimaging data bits are encoded into symbols which have, say, twice the duration of the corresponding encoded TV bits. It is not, however, our objective to weigh the relative merits of these techniques against those of our concatenation scheme.

II. Description of the Concatenation Scheme

Figure 1 is a gross block diagram of the Viterbi convolutional coded telemetry system which is to be used on MJS. The output of the Viterbi decoder is an imperfect, time-delayed version of the output of the data source. According to Table 1, this system requires 0.3–0.8 less energy per transmitted bit to achieve a bit error probability of 0.005 than does the (32, 6) biorthogonal coding system.

Now let us regard the encoder–channel–decoder ensemble of Fig. 1 as a single block, and call it the *superchannel*. From this viewpoint, the superchannel is a block which accepts binary data and expels a noisy version of those data. The idea of concatenated coding, as it applies

to our problem, is to combat the noise of the superchannel with coding; i.e., to insert a suitably chosen encoder–decoder pair into the block diagram of Fig. 1, as shown in Fig. 2.

It remains, of course, to describe in detail the contents of the blocks labeled “superencoder” and “superdecoder.” The code to be used on the Viterbi superchannel cannot be expected to be a simple one, since the statistics of that channel are very complicated. Indeed, no satisfactory mathematical model for this channel exists at present, and so, in order to verify that a particular coding scheme actually performs adequately, extensive computer simulation is necessary. Basically, however, what is needed is a code which is capable of correcting relatively infrequent bursts of errors, since experience has taught us that, at a bit signal-to-noise ratio adequate to drive the Viterbi decoder’s bit error probability down to 5×10^{-3} , the errors, when they do occur, cluster in bursts of length 10–20 or so.

As mentioned in the introduction, we propose an interleaved version of the (24, 12) Golay code. If the “depth of interleaving” (an integer defined precisely below) is denoted by k , then the supercode will be a $(24k, 12k)$ block code. That is, the nonimaging data stream will be partitioned into blocks of size $12k$, and to each such block will be adjoined an additional $12k$ parity-check bits. Thus, the superencoder of Fig. 2 will emit binary codewords of length $24k$. The description of the actual implementation of the encoder will be deferred to Section V; here, it will suffice to point out that if a $24k$ -bit codeword is denoted by $(x_0, x_1, x_2, \dots, x_{24k-1})$, each of the k 24-bit subsequences $(x_0, x_k, x_{2k}, \dots, x_{23k})$, $(x_1, x_{k+1}, x_{2k+1}, \dots, x_{23k+1})$, \dots , $(x_{k-1}, x_{2k-1}, \dots, x_{24k-1})$ will turn out to be a codeword of the basic (24, 12) Golay code. Thus, the big $24k$ -bit codeword consists of k Golay codewords which are “interleaved.”

Now the (24, 12) Golay code, when decoded optimally, is capable of correcting any pattern of three or fewer errors, and many patterns of four errors. This fact makes the overall $(24k, 12k)$ an extremely powerful code for correcting multiple bursts of bit errors: any pattern of bit errors in a block of length $24k$ can be completely corrected so long as none of the k 24-bit Golay codewords out of which the overall codeword has been constructed contains more than three of the bit errors. Thus, in particular, if all the bit errors are confined to a “burst” of length $3k$ or less, no individual Golay codeword can contain more than three of the errors, and so all of the errors will be corrected. Of course, many other more complex patterns of bit errors will be corrected also.

Thus, the Viterbi decoder produces many complex combinations of bursts of errors, and the interleaved Golay code is capable of correcting many complex combinations of bursts of errors. The next section presents quantitative verification that the code and the channel are indeed well matched.

III. Simulation Results

We have simulated the performance of our concatenation scheme with the aid of bit error statistics obtained from a commercially manufactured (7, 1/2) Viterbi decoder, and from a software (6, 1/3) Viterbi decoder. The (7, 1/2) statistics were stored on two reels of magnetic tape. One reel contained 1.3×10^8 bits which were decoded at an E_b/N_0 of 2.5 dB, and a resulting bit error probability of 6.2×10^{-3} . The other reel had 9.9×10^7 bits, $E_b/N_0 = 2.0$ dB, $P_E(\text{bit}) = 1.8 \times 10^{-2}$. The (6, 1/3) statistics were stored on one reel of magnetic tape containing 6.4×10^6 bits decoded with $E_b/N_0 = 1.9$ dB, $P_E(\text{bit}) = 5 \times 10^{-3}$, and 6.4×10^6 bits with $E_b/N_0 = 1.5$ dB, $P_E(\text{bit}) = 10^{-2}$.¹ The interleaving depths $k = 12, 16, 20$, and 24 were all tested.

In addition, four different versions of the decoding algorithm were tested. The reason we did not settle on one decoding algorithm is that the Golay code is, in fact, a (23, 12) rather than a (24, 12) code. The 24th bit of each codeword is merely an overall parity-check bit. And while, for the (23, 12) code, an algorithm which corrects any pattern of three or fewer errors can neither correct nor detect any error pattern of more than three errors, the extended (24, 12) code has the ability to detect and correct many patterns of four errors. It was our desire to test several combinations of detection and correction which are possible with the extended code.

The four decoding rules will be denoted by A1, A2, B1, B2. Roughly speaking, A denotes an algorithm which only corrects patterns of up to three errors, and B denotes an algorithm which corrects not only all patterns of three or fewer errors but also many likely patterns of four errors. The algorithms labeled 1, when confronted with an error pattern that cannot be corrected, make no change in the received bits but merely pass them along un-

processed. Those labeled 2 "erase" all the bits in a codeword which appears to have suffered an uncorrectable error pattern. Details of these algorithms will be given in the next section; we conclude this section with performance tables of algorithms A1, A2, B1, and B2 for the interleaving depths $k = 12, 16, 20$, and 24.

There are two sets of numbers for each algorithm: those labeled (0.0062, 0.005) and those labeled "(0.018, 0.01)". The first entry in a pair under a (0.0062, 0.005) rubric represents an error or erasure probability for the concatenation scheme combined with the (7, 1/2) Viterbi decoder with bit error probability 6.2×10^{-3} ; the second entry gives the same probability for the scheme combined with the (6, 1/3) decoder with a raw bit error probability of 5×10^{-3} . The entries under (0.018, 0.01) have the same significance for the higher bit error probabilities 1.8×10^{-2} for the (7, 1/2) code and 1.0×10^{-2} for the (6, 1/3) code. A condensed exponential notation is used; e.g., 1.7E-2 denotes 1.7×10^{-2} . Error probability is denoted by P_E and erasure probability by P_{ER} . Probabilities marked with an asterisk (*) were computed from less than 10 error events and so may not be reliable.

Algorithm A1			
	k	$P_E(\text{bit})$	$P_E(\text{block})$
(0.0062, 0.005)	12	(1.8E-4, 1.4E-4)	(5.8E-3, 2.9E-3)
	16	(6.9E-5, 4.7E-5)	(3.4E-3, 2.1E-3)
	20	(3.3E-5, 1.7E-5)	(2.2E-3, 1.5E-3)
	24	(1.9E-5, 8.1E-6)	(1.7E-3, 9.0E-4)
(0.018, 0.01)	12	(1.5E-3, 6.8E-4)	(3.6E-2, 1.2E-2)
	16	(8.0E-4, 3.4E-4)	(3.0E-2, 1.0E-2)
	20	(4.9E-4, 1.6E-4)	(2.5E-2, 8.4E-3)
	24	(3.5E-4, 8.0E-5)	(2.4E-2, 6.3E-3)
Algorithm B1			
	k	$P_E(\text{bit})$	$P_E(\text{block})$
(0.0062, 0.005)	12	(1.0E-4, 6.7E-5)	(2.5E-3, 1.4E-3)
	16	(3.9E-5, 2.3E-5)	(1.5E-3, 9.6E-4)
	20	(2.3E-5, 8.7E-6)	(1.2E-3, 4.5E-4*)
	24	(1.4E-5, 4.4E-6)	(1.0E-3, 3.6E-4*)
(0.018, 0.01)	12	(1.0E-3, 4.3E-4)	(2.1E-2, 7.3E-3)
	16	(6.0E-4, 1.9E-4)	(1.8E-2, 5.7E-3)
	20	(3.9E-4, 1.0E-4)	(1.7E-2, 5.1E-3)
	24	(3.1E-4, 5.5E-5)	(1.8E-2, 3.8E-3)

¹There is a small discrepancy between these figures and those of Table 1 because it is not feasible to implement maximum-likelihood survivor selection at high bit rates. The faster, suboptimum Viterbi decoder is slightly less efficient. The reader interested in details should consult Ref. 1.

Algorithm A2

	k	P_E (bit)	P_E (block)	P_{ER} (block)
(0.0062, 0.005)	12	(1.1E-5, 7.5E-6)	(2.8E-4, 1.8E-4*)	(5.5E-3, 2.7E-3)
	16	(2.5E-6, <E-7*)	(1.0E-4, <E-5*)	(3.3E-3, 2.1E-3)
	20	(1.3E-6, <E-7*)	(6.9E-4, <E-5*)	(2.2E-3, 1.5E-3)
	24	(6.6E-7, <E-7*)	(4.8E-5, <E-5*)	(1.7E-3, 9.0E-4)
(0.018, 0.01)	12	(7.9E-5, 1.4E-5)	(2.1E-3, 3.6E-4*)	(3.5E-2, 1.2E-2)
	16	(3.6E-5, 1.3E-6*)	(1.4E-3, 6.0E-5*)	(2.8E-2, 1.0E-2)
	20	(1.8E-5, 5.0E-6)	(9.4E-4, 3.0E-4*)	(2.4E-2, 8.1E-3)
	24	(1.3E-5, 1.3E-6*)	(9.1E-4, 9.0E-5*)	(2.3E-2, 6.2E-3)

Algorithm B2

	k	P_E (bit)	P_E (block)	P_{ER} (block)
(0.0062, 0.005)	12	(5.0E-5, 4.1E-5)	(1.3E-3, 9.0E-4)	(1.2E-3, 4.9E-4)
	16	(2.1E-5, 1.1E-5)	(7.5E-4, 3.6E-4*)	(7.1E-4, 6.0E-4)
	20	(1.2E-5, 3.7E-6)	(5.8E-4, 1.5E-4*)	(6.3E-4, 3.0E-4)
	24	(6.2E-6, 2.5E-6)	(4.1E-4, 1.8E-4*)	(6.3E-4, 1.8E-4)
(0.018, 0.01)	12	(3.6E-4, 1.6E-4)	(9.4E-3, 3.3E-3)	(1.2E-2, 4.0E-3)
	16	(2.0E-4, 5.3E-5)	(7.3E-3, 2.0E-3)	(1.1E-2, 3.8E-3)
	20	(1.3E-4, 4.6E-5)	(6.1E-3, 2.1E-3)	(1.1E-2, 3.0E-3)
	24	(1.1E-4, 1.9E-5)	(6.4E-3, 1.3E-3)	(1.1E-2, 2.6E-3)

IV. Conceptual Description of the Decoding Algorithm

All of our decoding algorithms for the (24, 12) extended Golay code can be summarized as follows:

- (1) Accept new 24-bit word.
- (2) Calculate the 12-bit syndrome s for the received 24-bit word.
- (3) Find the entry opposite s in the (previously calculated) syndrome table.
- (4) If this entry is a 24-bit error pattern, add (mod 2) this pattern to the received word, and output the resulting 24-bit word. If the entry is the special symbol *, output the received word unprocessed for algorithms A1 and B1, or erase all 24 bits of the received word for algorithms A2 and B2.
- (5) Go to step 1.

The above skeleton completely describes the algorithms, except for the syndrome table. The rest of this section will be devoted to a description of this table. We

will begin with an outline of the theory of decoding a general linear code, and then specialize to the extended Golay code.

Any (n, k) linear code can be characterized as the set of binary n -tuples $c = (c_0, c_1, \dots, c_{n-1})$ which satisfy the matrix equation

$$cH = 0$$

where H is a fixed $n \times (n - k)$ binary matrix of rank $n - k$. Now we define the *syndrome* s for an arbitrary binary vector v of length n by the equation

$$vH = s \tag{1}$$

The syndrome s is an $n - k$ dimensional binary vector. For a fixed syndrome s , the set of vectors v which satisfy (1) is called a *coset* of the code C ; each such coset contains 2^k n -dimensional vectors. It turns out that if a codeword c is transmitted over a noisy channel, and if v is received, the error pattern, i.e., the difference $v - c$, must lie in the coset defined by (1). For any coset of the code, certain vectors in that coset are at least as likely to turn up as

error patterns as any of the others. In such a case, one of these likely vectors may be designated as a *coset leader*; in most applications, the coset leader is chosen from among the vectors of least weight in the coset. In the decoding algorithm, when a vector is received whose coset has a leader, that leader is added modulo 2 to the received vector, and the resulting vector is the output of the decoder. If the received vector belongs to a coset without a designated leader, the received bits are either unprocessed or are erased by the decoder. These two alternative ways of handling the leaderless cosets correspond to the algorithms of types 1 and 2 described in the last section.

To complete the description of our decoding algorithms for the (24, 12) extended Golay code, then, we must indicate how the coset leaders were chosen.

The first step was to ensure that all error patterns of weight 0, 1, 2, or 3 would be corrected. Thus, each of these

$$\binom{24}{0} + \binom{24}{1} + \binom{24}{2} + \binom{24}{3} = 2325$$

low-weight error patterns was designated as a coset leader. (It is a fundamental fact about the Golay code that no two of these vectors lie in the same coset.) For the algorithms labeled A, no other cosets were given leaders.

For the algorithms labeled B, however, we used some of the remaining 1771 cosets to correct certain common error patterns of weight 4. Since

$$1771 = \frac{1}{6} \binom{24}{4}$$

it is clear that we could hope to correct at most one-sixth of all patterns of weight 4. Indeed, each of the 1771 remaining cosets contains exactly six vectors of weight 4, and those six vectors are "disjoint" in the sense that their modulo-2 sum is 11111111111111111111. Fortunately, however, it turns out that among the 10626 weight-4 error patterns that appear in the interleaved Golay code in the presence of "Viterbi noise," only a small fraction occur with substantial probability. These are the "tame" patterns, in which the four ones are confined to one or two short bursts; i.e., the 651 patterns of the form ...1...111..., ...111...1..., and ...11...11...

For example, for the twelve-fold interleaved Golay code, and the tape of (7, 1/2) Viterbi noise at a bit error probability of 6.2×10^{-3} , 4030 of the 5529504 processed Golay codewords suffered exactly four bit errors, and 3187 of these were of the tame variety described above. (In fact,

pattern 1111 occurred 1322 times, 33% of the total; and the tame patterns of overall burst length ≤ 7 accounted for 58% of the total.)

Thus, in the B algorithms, we wanted as many of the tame weight-4 error patterns to be coset leaders as possible. This was done (by computer) by generating the 651 tame patterns sequentially, in order of increasing burst length, computing the syndromes, and making the pattern a coset leader if its coset was leaderless at the time it was being examined. It turned out that we could make 542 of the 651 tame patterns coset leaders. (The shortest tame pattern which belongs to the same coset as an even shorter tame pattern, and so cannot be corrected, is 1000111. Thus, the probability of a tame pattern being corrected is substantially higher than $542/651 = 0.833$).

Finally, a further study of the weight-4 error patterns failed to reveal any particular class of patterns which occurred a disproportionate number of times, and so we chose no other coset leaders. Hence, in the B algorithms, 2867 of the cosets had leaders, and 1229 did not.

V. Implementation—Encoding

It is possible to take advantage of the fact that the basic (23, 12) Golay code is cyclic to design a very simple encoding circuitry for the k -fold interleaving of the extended (24, 12) Golay code. Figure 3 is a diagram of one possible encoding configuration. The $12k$ data bits are sent down the channel and into the shift register, with the four switches in the "up" position as shown. The switches are then put in the "down" position, the $12k$ parity-check bits are sent down the channel, and the shift register is filled with zeros. The switches are then returned to the "up" position, and the transmission of a new block begins.

For k in the range 12–24, the circuit of Fig. 3 could be built with 20 or so standard metal oxide semiconductor (MOS) integrated circuits at a cost of less than \$100; alternatively, it could be put on one specially ordered integrated circuit, although this would increase the cost considerably.

Finally, let us note one very important feature of the encoder of Fig. 3: Although the overall $24k$ -bit codewords consist of k interleaved 24-bit codewords, *the data bits are not interleaved* but appear serially in the encoded stream in blocks of $12k$ separated by blocks of $12k$ parity-check bits.

VI. Implementation—Decoding

The decoding algorithms we used were software algorithms, written for an SDS 930 computer. However, we shall argue at the end of this section that at least some of the computation would be better done with relatively inexpensive special-purpose hardware.

We assume that the $24k$ -bit word which is to be decoded has somehow been loaded into the computer. The decoder's first task is to "de-interleave" the k component Golay codewords. Thus, if $(y_0, y_1, y_2, \dots, y_{24k-1})$ is the received word, the first decoding step is to arrange the bits into k 24-bit words, as follows:

$$\begin{array}{ll} \text{Word 1} & y_0, y_k, y_{2k}, \dots, y_{23k} \\ \text{Word 2} & y_1, y_{k+1}, y_{2k+1}, \dots, y_{23k+1} \\ & \vdots \\ \text{Word } k & y_{k-1}, y_{2k-1}, y_{3k-1}, \dots, y_{24k-1} \end{array}$$

(Notice that at this stage, the data bits have been scrambled.) Next, each of the k 12-bit words is decoded, as indicated in Section IV; i.e., the syndromes are calculated, and the error patterns (coset leaders) corresponding to each syndrome are added modulo 2 to the received words. For the B algorithms, this will require a stored table of 4096 12-bit words (12 rather than 24, because the last 12 bits of each Golay codeword are not carrying information, only noise-combating parity). However, for the A algorithms, we need only a syndrome table of 2048 12-bit words, since a word from the $(24, 12)$ code is merely a word from the $(23, 12)$ code to which has been appended an overall parity check. Thus, every codeword in the $(24, 12)$ code has even weight (i.e., an even number of ones), and so the parity of the number of errors that have occurred in a 24-bit word is the same as the parity of the number of ones in that word. This leads us to the following modified version of the decoding algorithms:

- A1. Compute the 11-bit syndrome for the first 23 bits, regarded as a noisy version of a codeword from the $(23, 12)$ code.
- A2. Find the 23-bit error pattern of weight 0, 1, 2, or 3 corresponding to the syndrome calculated in A2.
- A3. If the error pattern found in A3 has the same parity as the received 24-bit word, add the error pattern to the first 23 bits. Otherwise, the decoding algorithm has detected four or more errors.

This version of the algorithm requires the storage of only 2048 13-bit words (12 bits of the error pattern and one

extra bit which gives the parity of the complete 23-bit error pattern). Thus, if computer storage is at a premium, the A algorithms enjoy a considerable advantage over the B algorithms.

In any event, when the k 24-bit words have been decoded, the result will be k blocks of corrected data:

$$\begin{array}{ll} \text{Corrected word 1} & X_0, X_k, \dots, X_{11k} \\ \text{Corrected word 2} & X_1, X_{k+1}, \dots, X_{11k+1} \\ & \vdots \\ \text{Corrected word } k & X_{k-1}, X_{2k-1}, \dots, X_{12k-1} \end{array}$$

Each X_i is the decoder's estimate of the corresponding transmitted bit X_i , or a special erasure symbol. The final decoding step is the re-interleaving of the k words to get the corrected data stream $X_0, X_1, \dots, X_{12k-1}$.

The program we have written to do this decoding on the SDS 930 reads the incoming codewords from magnetic tape, and can process the data at a rate of 6000 (information) bits/s. This rate is well in excess of the currently estimated nonimaging science encounter data rate of 4500 bits/s.

Although current plans call for software implementation of the decoding algorithm, we cannot conclude this report without making some mention of the advantages of hardware decoding. We have found that over 70% of the decoding time in our program is devoted to the de-interleaving and re-interleaving of the k 24-bit Golay codewords, and that most of the remainder of the decoding time is consumed in computing the k 12-bit syndromes. Both of these operations are quite awkward in most assembly languages but could be implemented almost trivially with a handful of standard MOS integrated circuits. If outboard circuitry for performing these two functions became available, the entire decoding algorithm would make a negligible demand on the computer's central processing unit (CPU) time. The only remaining burden would be the large table of 2048 or 4096 13- or 12-bit words. This table could easily be stored on a special-purpose read-only memory (ROM), making the entire decoder disjoint from the main computer.

However, a final decision on the software-hardware question must be deferred until a careful study of the various tradeoffs (available computing power, reliability, expense, etc.) involved can be made.

Reference

1. Gilhousen, K. S., Heller, J. A., Jacobs, I. M., and Viterbi, A. J., *Coding Systems Study for High Data Rate Telemetry Links*, NASA CR 114278, prepared by Linkabit Corporation, 1972.

Table 1. E_b/N_0 (in dB) required to produce certain bit error probabilities for four telemetry systems

Bit error probabilities	Uncoded	(32, 6) biorthogonal	(7, 1/2) convolutional	(6, 1/3) convolutional
0.01	4.3	2.3	2.3	1.7
0.005	5.2	2.9	2.6	2.1
0.001	6.8	4.0	3.2	2.7

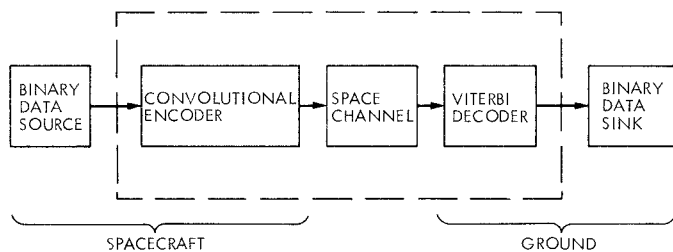


Fig. 1. Viterbi convolutional coded telemetry system for MJS

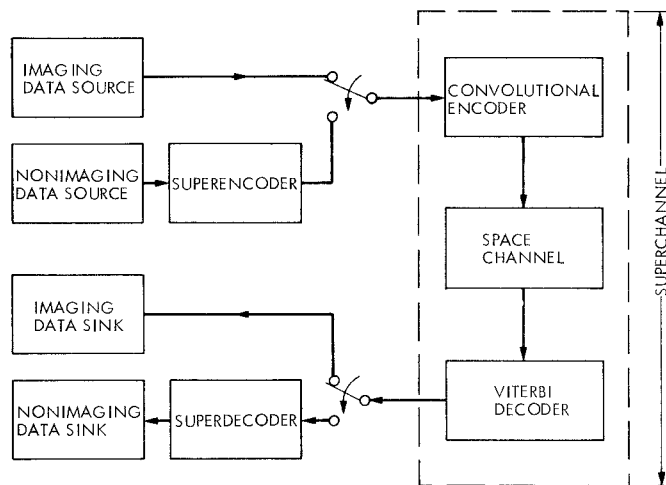


Fig. 2. Viterbi system with encoder-decoder pair

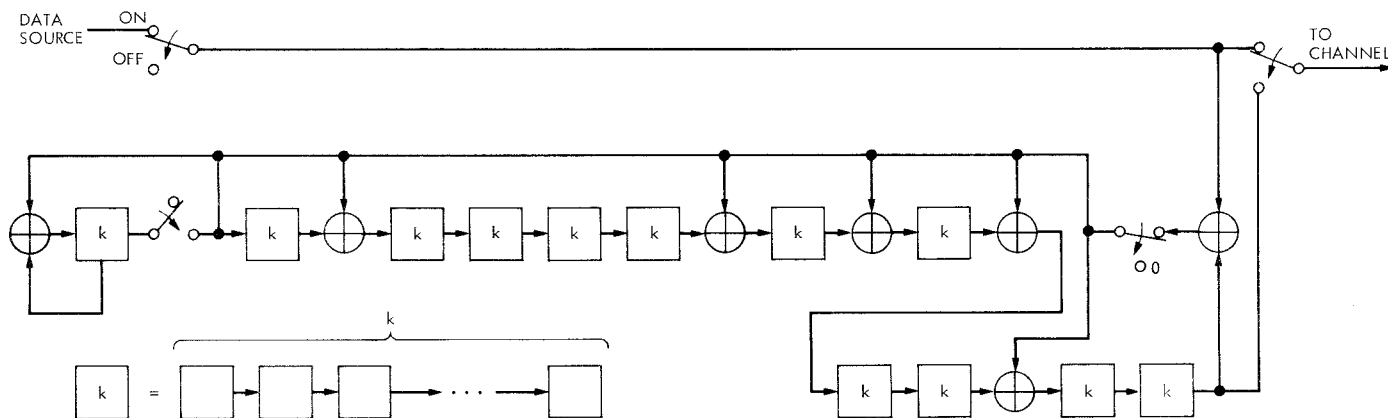


Fig. 3. Encoder for k -fold interleaving of Golay (24, 12) code